CONF - 7810149--1

LA-UR-78-2813

MASTER

TITLE: SIMPLIFIED SOLUTION ALGORITHMS FOR FLUID FLOW PROBLEMS

AUTHOR(S): C. W. Hirt

SUBMITTED TO: Numerical Methods for Partial Differential Equations Seminar, University of Wisconsin, 23 Oct 1978

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87545

An Affirmative Action/Equal Opportunity Employer

Form No. 836
St. No. 2629
1/75

UNITED STATES
ENERGY RESEARCH AND
DEVELOPMENT ADMINISTRATION
CONTRACT W-7405-ENG. 36

ABSTRACT

A simplified algorithm is described for the numerical solution of the Navier-Stokes equations. Because of its simple construction, the algorithm serves as a good introduction to numerical fluid dynamics as well as a basis for developing many kinds of new solution methods. To illustrate the flexibility of this algorithm simple modifications are described for introducing internal obstacles, an accelerated steady-state solution method, a potential flow option, and a method of increasing numerical accuracy.

I. INTRODUCTION

There are many advocates and practitioners of numerical fluid dynamics. There are also nearly as many numerical methods or codes. Most of these codes differ only in matters related to choices for finite difference approximations, boundary condition options, special purpose features, or other details. When stripped to their essentials the majority of solution methods reduce to relatively simple algorithms.

In this lecture we first present such a stripped-down algorithm for the numerical solution of the dynamics of incompressible, Navier-Stokes fluid. This solution algorithm (SOLA) is simple, straightforward, and provides a basis for learning the essential elements needed to obtain numerical solutions [1].

We shall then look at a variety of modifications and extensions of the basic algorithm. For example, easy ways of increasing accuracy, achieving fast steady-state solutions, adding a potential flow option, or including internal obstacles. In addition to the modifications described here, there are extended versions of SOLA available for treating free-boundary problems (SOLA-SURF) [1], and compressibility effects (SOLA-ICE) [2].

The presentation of a variety of modifications that may be made to the SOLA code serves a dual purpose. Each modification focuses attention on some element of the basic algorithm and its relationship to the remaining elements. These modifications also illustrate how easy it is to develop new and powerful computational schemes for many different applications.

The SOLA code described here is publically available from the National Energy Software Center (formerly the Argonne Code Center), 9700 South Cass Avenue, Argonne, IL 60439.

II. SOLA-A SOLUTION ALGORITHM FOR INCOMPRESSIBLE FLUID FLOW

The solution algorithm (SOLA) is a simplified version of the Marker-and-Cell (MAC) method originally developed by Harlow, et al. [3]. It is a numerical method for the solution of the time-dependent, two-dimensional, Navier-Stokes equations,

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} + \xi \frac{u^2}{x} = - \frac{\partial p}{\partial x} + g_x + \nu \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \xi \left(\frac{1}{x} \frac{\partial u}{\partial x} - \frac{u}{x^2} \right) \right] \tag{1}$$

$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} + \xi \frac{uv}{x} = - \frac{\partial p}{\partial y} + g_y + \nu \left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\xi}{x} \frac{\partial v}{\partial x} \right]$$

and the incompressibility condition

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \xi \frac{u}{x} = 0 \quad . \tag{2}$$

Here the velocity components (u,v) are in the coordinate directions (x,y), p is the ratio of pressure to constant density, (g_x, g_y) are body accelerations, and ν is a constant coefficient of kinematic viscosity. The parameter ξ is set to zero when calculations are to be performed in Cartesian coordinates. By setting ξ equal to unity the equations are those for cylindrical coordinates in which x is the radial direction and y the axial direction.

The basic solution technique contained in SOLA provides solutions of Eqs. (1-2) in a rectangular region whose boundaries may be specified in various ways through the selection of input parameters. In particular, options are available for rigid walls with free-slip or no-slip tangential velocity conditions, as continuative outflow boundaries, or as periodic boundaries. Constant pressure or specified inflow and outflow boundaries are also easily added, as are internal obstacles, sources, and sinks.

A. Numerical Approximations

The finite-difference mesh used in SOLA, see Fig. 1, consists of rectangular cells of width δx and height δy. The mesh region containing fluid is composed of IBAR cells in the x-direction, labeled with the index i, and JBAR cells in the y-direction, labeled with the index j. The fluid region is surrounded by a single layer of fictitious cells so that the complete mesh consists of IBAR+2 by JBAR+2 cells. The fictitious cells are used to set boundary conditions so that the same difference equations used in the interior of the mesh can also be used at the boundaries.

Fluid velocity components and pressures are located at staggered cell positions as shown in Fig. 2. This staggering has been chosen to simplify the difference approximations to Eqs. (1-2).

Subscripts are used to denote cell locations and superscripts for the time level at which quantities are evaluated. For example, $u^n_{i+\frac{1}{2},j}$ denotes the u-velocity at time $n\delta t$ located at the right side of cell (i,j). Using this notation the finite-difference equations used to approximation the Navier-Stokes equations, Eqs. 1, have the form,

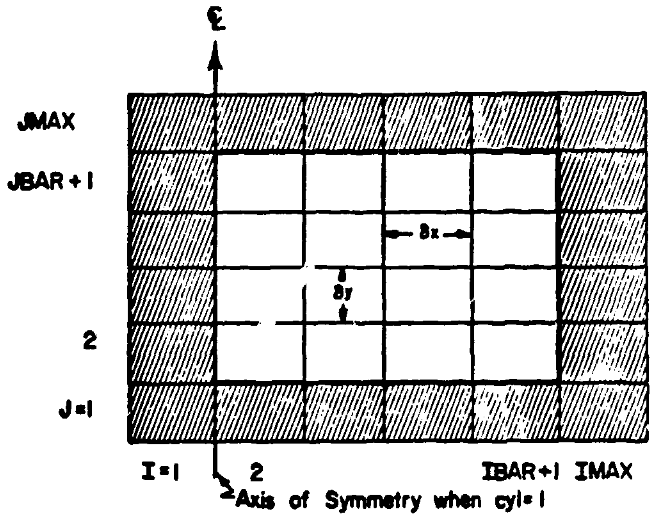

Fig. 1. General mesh arrangement, with fictitious boundary cells shaded.

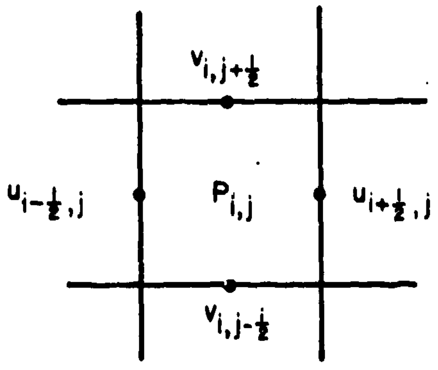

Fig. 2. Arrangement of finite difference variables in a typical mesh cell.

$$u^{n+1}_{i+\frac{1}{2},j} = u^{n}_{i+\frac{1}{2},j} + \delta t \left[\left(p^{n}_{i,j} - p^{n}_{i+1,j}\right)/\delta x + g_x - \mathrm{FUX} - \mathrm{FUY} - \mathrm{FUC} + \mathrm{VISX}\right]$$

$$v^{n+1}_{i,j+\frac{1}{2}} = v^{n}_{i,j+\frac{1}{2}} + \delta t \left[\left(p^{n}_{i,j} - p^{n}_{i,j+1}\right)/\delta y + g_y - \mathrm{FVX} - \mathrm{FVY} - \mathrm{FVC} + \mathrm{VISY}\right] , \quad (3)$$

where, e.g., FUX represents the expression used for the convective flux of u in the x-direction, and VISX represents the expression used for the viscous acceleration of u. All terms on the right side of Eq. (3) are evaluated using known time level n quantities.

As far as the basic SOLA algorithm is concerned, the difference expressions chosen for the convective and viscous accelerations in Eq. 3 are immaterial, provided the resulting equations lead to numerically stable approximations. Thus, the user could readily insert other difference approximations for FUX, VISX, etc., without having to change the remainder of the algorithm (except possibly for boundary condition changes needed to be consistent with the new expressions).

In the publically available version of SOLA a combination of "central" and "donor-cell" differencing is used for the convective fluxes. For example, FUY is approximated by the expression,

$$\mathrm{FUY} = \frac{1}{\delta y}\Big[v_{i+\frac{1}{2},j+\frac{1}{2}}\, u_{i+\frac{1}{2},j+\frac{1}{2}} + \alpha\left|v_{i+\frac{1}{2},j+\frac{1}{2}}\right|\left(u_{i+\frac{1}{2},j} - u_{i+\frac{1}{2},j+1}\right) - v_{i+\frac{1}{2},j-\frac{1}{2}}\, u_{i+\frac{1}{2},j-\frac{1}{2}} - \alpha\left|v_{i+\frac{1}{2},j-\frac{1}{2}}\right|\left(u_{i+\frac{1}{2},j-1} - u_{i+\frac{1}{2},j}\right)\Big] .$$

Simple averages are used for quantities needed at locations where they are not defined, e.g., $u_{i+\frac{1}{2},j+\frac{1}{2}} = 1/2\ (u_{i+\frac{1}{2},j} + u_{i+\frac{1}{2},j+1})$. The parameter α is a user specified input constant, whose value is between zero and one. When α is zero the convective approximations are centered in space, but when α equals one the approximations use the upstream or donor-cell values of the quantities to be fluxed. Unfortunately, the centered form leads to equations that are computationally unstable [4]. In general, α should be chosen slightly larger than the maximum value occurring in the mesh of $|\frac{u\delta t}{\delta x}|$ or $|\frac{v\delta t}{\delta y}|$.

All other convective flux contributions are approximated in SOLA in a fashion similar to FUY. The viscous accelerations are approximated by central difference expressions. For a complete set of difference equations, Ref. 1 should be consulted.

The velocities computed according to Eqs. (3) will not, in general, satisfy the condition of incompressibility. This condition, Eq. (2), for a typical cell (i,j) is approximated as

$$\frac{1}{\delta x}\left(u^{n+1}_{i+\frac{1}{2},j} - u^{n+1}_{i-\frac{1}{2},j}\right) + \frac{1}{\delta y}\left(v^{n+1}_{i,j+\frac{1}{2}} - v^{n+1}_{i,j-\frac{1}{2}}\right) + \frac{\xi}{2\delta x\,(i-1.5)}\left(u^{n+1}_{i+\frac{1}{2},j} + u^{n+1}_{i-\frac{1}{2},j}\right) = 0 \quad . \tag{4}$$

To satisfy this condition the pressure in cell (i,j) is suitably changed. For example, when the velocity divergence is negative, corresponding to a net flow of fluid into the cell, the pressure is increased to prevent the inflow. When the divergence is positive, corresponding to a net outflow, the pressure is reduced to prevent the outflow. If D is the velocity divergence, then the pressure change needed to drive D to zero is

$$\delta p = -\omega D/\left[2\delta t\left(\frac{1}{\delta x^2} + \frac{1}{\delta y^2}\right)\right] , \tag{5}$$

where ω is an over-relaxation parameter ($1 < \omega < 2$). Once δp is determined, the cell pressure is updated to $p_{i,j} + \delta p$ and the four cell edge velocities are also updated to reflect this change,

$$\begin{aligned} u_{i\pm\frac{1}{2},j} &\rightarrow u_{i\pm\frac{1}{2},j} \pm \delta t\ \delta p/\delta x \\ v_{i,j\pm\frac{1}{2}} &\rightarrow v_{i,j\pm\frac{1}{2}} \pm \delta t\ \delta p/\delta y \quad , \end{aligned} \tag{6}$$

The pressure adjustments must be done iteratively, because a change in one cell will upset the b ance in neighboring cells. Convergence is achieved when all cells have D magnitudes less than some small predetermined value. It can be easily shown [5] that this iterative pressure adjustment is equivalent to solving a Poisson equation for the pressure.

To summarize the above steps, which make up a complete computational cycle:

(1) Approximate new velocities are computed from the explicit difference equations, Eq. (3).

(2) These velocities and cell pressures are then iteratively adjusted to satisfy the incompressibility condition, Eq. (4).

(3) Finally the time is advanced to $t+\delta t$ and the new pressure and velocities may be used as starting values for the next cycle of computation. Bookkeeping and output are also done in this step as desired.

B. Boundary Conditions

To complete the basic SOLA method we must specify boundary conditions. For convenience the code has four boundary condition options that may be selected through input parameters. These options are rigid free-slip and rigid no-slip walls, continuative outflow boundaries, and periodic boundaries.

All boundary conditions are imposed by suitably defining flow variables in the fictitious boundary cells. For example, consider the left boundary:

(1) For a rigid, free-slip wall,

$$u_{3/2,j} = 0.0, \quad v_{1,j+\frac{1}{2}} = v_{2,j+\frac{1}{2}}$$

(2) For a rigid, no-slip wall,

$$u_{3/2,j} = 0, \quad v_{1,j+\frac{1}{2}} = - v_{2,j+\frac{1}{2}}$$

(3) For a continuative boundary,

$$u_{3/2,j} = u_{5/2,j}, \quad v_{1,j+\frac{1}{2}} = v_{2,j+\frac{1}{2}}$$

(4) For x-periodic boundaries, on the left

$$u_{3/2,j} = u_{IBAR+\frac{1}{2},j}\ ,\ v_{1,j+\frac{1}{2}} = v_{IBAR,j+\frac{1}{2}}$$

$$v_{2,j+\frac{1}{2}} = v_{IBAR+1,j+\frac{1}{2}}\ ,\ p_{2,j} = p_{IBAR+1,j}$$

and on the right

$$u_{IBAR+3/2,j} = u_{5/2,j}\ ,\ v_{IBAR+2,j+\frac{1}{2}} = v_{3,j+\frac{1}{2}}\ .$$

In addition to the above boundary conditions the code has a special section reserved where additional conditions can be imposed. In all cases the additional conditions override the standard ones. For example, specified inflow or outflow boundaries are generated by setting the fictitious cell and boundary velocities to the desired values. For internal obstacles with shapes constructed by blocking out mesh cells, we add in the special boundary condition section statements that set all velocities in the blocked out cells to zero.

C. Stability and Accuracy

To prevent numerical instabilities or inaccuracies, certain restrictions must be observed in defining the mesh increments δx and δy, the time increment δt, and the upstream differencing parameter α. For accuracy, the mesh increments must be chosen to resolve the expected spatial variations of all dependent variables. Once a mesh has been chosen, the choice of the time increment necessary for stability is governed by two restrictions. First, material cannot be allowed to convect through more than one cell in one time step, because the difference equations assume fluxes only between adjacent cells. Thus, it is necessary that δt satisfy,

$$\delta t < \min\left\{\frac{\delta x}{|u|}\ ,\ \frac{\delta y}{|v|}\right\} \tag{7}$$

for every cell in the mesh. Usually, δt is chosen equal to 1/3 to 1/4 the minimum cell transit time. Second, when the kinematic viscosity is nonzero, momentum must not diffuse more than approximately one cell in one time step, or

$$\nu\ \delta t < 1/2\ \frac{\delta x^2\ \delta y^2}{\delta x^2 + \delta y^2} \tag{8}$$

when δt has been selected to satisfy the above accuracy and stability conditions, the parameter α is then chosen to satisfy

$$1 > \alpha > \max \left\{ \left|\frac{u\delta t}{\delta x}\right| \quad , \quad \left|\frac{v\delta t}{\delta y}\right| \right\} \quad . \qquad (9)$$

This last condition is needed to eliminate an instability that would otherwise develop because of the form chosen for the convective fluxes.

D. Summary

The basic solution algorithm described above is programmed in a straightforward and concise way in the SOLA code [1]. As it stands, it provides a powerful tool for the solution of many interesting flow problems. One of the most powerful aspects of SOLA, however, is the ease with which it can be modified or extended to handle new problems. Several examples of this are outlined in the following sections, which also include sample calculations illustrating a variety of possible applications.

III. MODIFICATIONS OF THE BASIC ALGORITHM

A. Additional Boundary Conditions

It has already been mentioned that additional exterior and interior boundary conditions, beyond those already built into the code, are easily added. To illustrate this capability let us consider the flow generated in the vicinity of an abrupt pipe expansion, Fig. 3. A cylindrical mesh is used that consists of 10+2 cells in the radial (x) direction and 25+2 cells in the axial (y) direction. At the upstream, or input end of the pipe, a 5 by 5 block of cells at the outer radius has been defined as an obstacle region. This is accomplished by inserting into the special boundary condition section statements that set $u=v=0$ at all faces of the obstacle cells. The specified inflow velocity at the bottom of the mesh is also defined in this section as a positive unit v-velocity and zero u-velocity in the first 5 fictitious cells at the bottom of the mesh. These values override the free-slip wall conditions set in the standard boundary condition section. Mesh-side boundaries were initialized as free-slip walls and the top was treated as a continuative boundary.

The velocity results shown in Fig. 3 after 400 cycles of time advancement, are stationary and show the presence of a large recirculation region existing downstream of the step expansion. The length of this region agrees well with available experimental data [6].

In an analogous way the user can introduce an almost unlimited variety of interior and exterior boundary conditions to fine obstacles, sources and sinks, and even the influence of flexible boundaries. In the latter case the boundary is treated as a specified normal velocity free-slip boundary. The velocities to be specified may come from a coupled structure code or other source. This method only works when the wall displacements are small compared to the

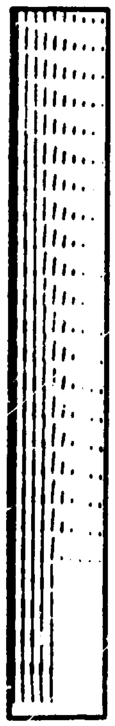

Fig. 3. Velocity field in region of a sudden pipe enlargement. All vectors start at cell centers. A vector length equal to horizontal cell size corresponds to a speed of 1/4 the inlet speed.

mesh cell size, for then the specification of the wall velocity at the original wall locations instead of its actual location, is a good approximation. Of course, all these boundary condition options are limited to boundaries that coincide with mesh cell boundaries. Modifications needed to treat more general shapes are considered in SOLA-SURF [1].

B. Steady-State Calculations

In many studies, like that described above, the main interest is in the asymptotically steady flow, and not in the details of the transients leading up to this flow. In these cases it has been shown by R. H. Hotchkiss that it is often possible to speed up the attainment of steady state by limiting the pressure iteration to only one iteration per time cycle. This is easily done by inputting in the code a large value for the convergence criterion, EPSI. The idea behind this technique is that wakes and other vorticity containing regions can only be generated by convective transport carrying vorticity into the flow from boundaries or other sources. While this is happening it is unnecessary to exactly satisfy the incompressibility condition. Once the flow reaches a steady state the incompressibility condition will be satisfied, because otherwise the one pass taken through the pressure iteration each cycle would alter the velocity and pressure fields, that is, the flow would not be steady.

In this way a considerable savings in computer time is usually realized by eliminating a large number of unnecessary pressure iterations. When this scheme is used, however, the over-relaxation parameter, ω, must not exceed unity, otherwise an instability may result.

An example of the use of this technique is provided by the abrupt expansion problem described in Sec. III.A. In the unmodified calculation, in which transients were computed accurately, it took approximately 23 sec of CDC-7600 computer time to reach steady state. With the above modification, steady state was reached in approximately 14 sec.

C. Potential Flow

Sometimes it is useful to have a potential flow solution to a particular problem. For those cases it isn't necessary to construct a new code, because SOLA can be easily modified to do the job. The basis for this modification comes from the observation that the finite-differenced momentum equations, Eqs. (3), can be cast into an approximation for the potential flow equations in which the velocity is equal to the gradient of a scalar potential. This is done by eliminating all body, convective, and viscous accelerations, and by setting the n-level velocities to zero in step one of each solution cycle so that what remains is

$$\begin{aligned} u^{n+1}_{i+\frac{1}{2},j} &= \frac{\delta t}{\delta x}\left(p^n_{i,j} - p^n_{i+1,j}\right) \\ v^{n+1}_{i,j+\frac{1}{2}} &= \frac{\delta t}{\delta y}\left(p^n_{i,j} - p^n_{i,j+1}\right) . \end{aligned} \tag{10}$$

Formally, we can identify δtp with a velocity potential. The incompressibility condition is still satisfied by iterating on the pressure as in the full SOLA code, and all boundary conditions may be used without modification. Thus, the only modification needed in SOLA to produce potential flow solutions is to bypass most of step one in the usual SOLA algorithm, such that Eqs. (3) are reduced to Eqs. (10).

In addition to being simple, this variation of the basic algorithm is quite instructive, for it emphasizes what is omitted from the full equations when the potential flow approximation is made. In particular, no information about the velocity field is retained from cycle to cycle, except for specified boundary velocities. If the boundary conditions are time independent, then only one solution cycle is needed to obtain the flow. No viscous effects may be included, and no flow features can be convected about, because these mechanisms have been omitted.

When SOLA is modified to have free surfaces, as in the SOLA-SURF code [1], this potential flow option can still be used, but then it is also necessary to modify the free-surface boundary condition for p, as described in Ref. 7.

The abrupt expansion problem used to illustrate the two previous modifications can also be used here. Figure 4 shows the velocity field generated when potential flow is assumed. The flow field is completely established in one

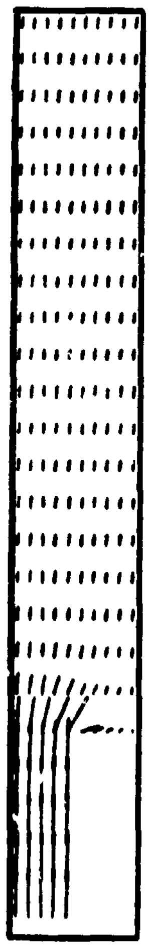

Fig. 4. Velocity field for potential flow at a region of sudden pipe enlargement. Compare with Fig. 3.

time step (requiring 983 iterations). Notice, however, that the recirculation region is entirely absent, because no vorticity is allowed to exist in the flow region.

D. Second Order Accurate Difference Approximations

Finite-Difference approximations used in the standard version of SOLA are first order accurate. That is, they have truncation errors proportional to the first power of the time increment, δt, and the first power of the space increments δx and δy. The advantage of these approximations is that they are simple and easy to keep computationally stable. For a great many applications they also provide accurate numerical solutions. In some cases, however, it is too costly to increase the number of cells to the point where the resolution is fine enough for accurate first order approximations. In these cases, it is often useful to have a second order accurate method.

In SOLA, second order accuracy can be quickly incorporated, without the introduction of additional storage arrays, by using a variant of a scheme employed by MacCormack [8]. The essence of this technique is best illustrated through application to the one-dimensional Burger's equation,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2} \quad . \tag{11}$$

A finite-difference approximation to Eq. (11) that is analogous to the approximations used in SOLA is

$$u_j^{n+1} = u_j^n + \delta t\ (-\mathrm{FUX} + \mathrm{VISX}) \tag{12}$$

where

$$\begin{aligned} \mathrm{FUX} &= \frac{1}{2\delta x}\left[u_j^n\left(u_{j+1}^n - u_{j-1}^n\right) - \alpha|u_j^n|\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right)\right] \\ \mathrm{VISX} &= \frac{\nu}{\delta x^2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) \ . \end{aligned} \tag{13}$$

Here, as in SOLA, $\alpha = 0$ results in a centered difference approximation that is spatially second order accurate, but also unconditionally unstable when $\nu = 0$. An α value of unity corresponds to upstream or donor cell differencing that is first order accurate and stable when

$$\frac{|u|\delta t}{\delta x} < 1 \quad .$$

A better understanding of the role α plays to produce a stable algorithm can be gained by checking how Eqs. (12)-(13) approximate Eq. (11). This is done by expanding the difference equation in a Taylor series about the point $j\delta x$ and $n\delta t$. Doing this we find the following differential approximation,

$$\frac{\partial u}{\partial t} + \left(u + \delta t u \frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial x} = \left(\nu + \frac{\alpha|u|\delta t}{2} - \frac{\delta t}{2} u^2 + 2\delta t \nu \frac{\partial u}{\partial x}\right) \frac{\partial^2 u}{\partial x^2}$$

$$+ \nu \delta t\, u \frac{\partial^3 u}{\partial x^3} - \frac{\delta t}{2} \nu^2 \frac{\partial^4 u}{\partial x^4} + O(\delta x^2, \delta x \delta t, \delta t^2). \tag{14}$$

In arriving at this result, a term involving $\partial^2 u/\partial t^2$ has been rewritten in terms of space derivatives by using the Taylor expanded equation itself. This replacement is justified, because the difference equation requires only one initial condition so its differential approximation should likewise require only one initial condition. The first term on the right side of Eq. (14) is a diffusion term. When $\nu = 0$ the remaining diffusion coefficient will be positive only when

$$\frac{\alpha|u|\delta x}{2} > \frac{\delta t u^2}{2} .$$

This result explains the SOLA rule-of-thumb stability requirement, $\alpha > |u\delta t/\delta x|$. When this condition is violated the difference equations yield exponentially growing solutions that are consequences of a negative diffusion coefficient.

To obtain second order accuracy we proceed as follows. First, compute an estimate for u_j^{n+1} using the available n-level quantities and full upstream ($\alpha = +1$) approximations. Denote this estimate by u_j^*,

$$u_j^* = u_j^n + \delta t\,[-\text{FUX} + \text{VISX}]^n_{\alpha=1} \quad . \tag{15}$$

Next, repeat this process by evaluating FUX and VISX using the u_j^* values and full downstream ($\alpha = -1$) approximations. Denote the new values by u_j^{**},

$$u_j^{**} = u_j^* + \delta t\,[-\text{FUX} + \text{VISX}]^*_{\alpha=-1} \quad . \tag{16}$$

Finally, the desired second order accurate values are given by

$$u_j^{n+1} = 1/2\left(u_j^n + u_j^{**}\right) . \tag{17}$$

That this is second order accurate can be seen by combining Eqs. (15)-(17),

$$u_j^{n+1} = \frac{1}{2} u_j^n + \frac{1}{2}\left\{u_j^* + \delta t\,[-\mathrm{FUX} + \mathrm{VISX}]^*_{\alpha=-1}\right\}$$
$$= u_j^n + \frac{1}{2}\,\delta t\left\{[-\mathrm{FUX} + \mathrm{VISX}]^n_{\alpha=1} + [-\mathrm{FUX} + \mathrm{VISX}]^*_{\alpha=-1}\right\} \quad . \tag{18}$$

Recalling that u_j^* values are first order estimates for u_j^{n+1}, the curly bracket is seen to contain an average of accelerations evaluated at levels n and n+1 and an average of accelerations evaluated with $\alpha=1$ and $\alpha=-1$. The net result after Taylor expanding is that all first order δt and first order α truncation errors cancel, leaving a second order accurate approximation.

It might appear from Eq. (17) that an additional storage array over the first order method is required in this scheme to accommodate the u_j^{**} values. This is not the case, however, if we rewrite the basic time advancement calculation as

$$\bar{u} = \beta f(u) + (1-\beta)\bar{u} \tag{19}$$

where f(u) represents the right side of Eq. (15) or (16). During the first pass through Eq. (19) we use $\beta=1$ and $u=u^n$, so that

$$u^* = \bar{u} = f(u^n) \quad .$$

Then we interchange storage arrays setting $u^* = u$ and $u=u^*$. During the second pass through Eq. (19) β is set equal to 1/2, and because of the interchanged arrays,

$$u^{n+1} = \bar{u} = \frac{1}{2} f(u^*) + \frac{1}{2} u^n \quad ,$$

which is equivalent to Eq.(17).

A list of FORTRAN statements that may be added to the basic SOLA code listed in Ref. 1, to give it this second order accurate option, can be obtained by writing directly to the author.

Because the second order method requires two passes through the convective and viscous acceleration calculations each cycle, calculation times are correspondingly larger than in the first order method. However, the increased accuracy means that larger space and time increments can often be used to reduce computation times. Unfortunately, it is not always easy to decide á priori when it is best to use a finely resolved, fast, first order calculation or a coarsely resolved, slow, second order calculation. In practice, a useful procedure is to use the simpler first order method for most calculations, but to check accuracy with an occasional second order calculation using the same mesh.

E. Additional Modifications

Numerous other features have been added to the SOLA algorithm at one time or another to achieve a variety of useful capabilities. For example, automatic time step controls, variable mesh increments (δx and δy), marker particles to trace flow patterns, a variable viscosity or turbulence model, and a coupled density equation for the study of stratified fluids.

Perhaps the most important extensions that have been made to SOLA are those contained in a set of codes also available from the National Energy Software Center. These codes are:

(1) SOLA-SURF, which has a free surface or rigid, curved surface capability. The curved surfaces are limited to configurations that are defined by their height above the bottom of the computational mesh in each column of cells.

(2) SOLA-ICE, which extends the incompressible algorithm in SOLA to compressible fluids, so that flows containing shock and rarefaction waves may be computed. Because of the implicit numerical formulation used in this code it can also be used for far subsonic (incompressible) flows.

In addition, there are several other SOLA codes soon to be installed in the Center. These consist of a two-dimensional code for two-phase flow analysis (SOLA-DF) [9], a code for two-phase flow in networks composed of one-dimensional components (SOLA-LOOP) [10], and a three-dimensional version with a free surface capability (SOLA-3D) [11].

From the examples described here, it should be obvious to the innovative user that these codes offer a basis for the development of an almost unlimited variety of new codes. In many cases the needed modifications can be made quickly and easily, because of the simple construction of the basic algorithms.

IV. Acknowledgments

The SOLA code series has resulted from the combined efforts of many members of Group T-3 of the Los Alamos Scientific Laboratory. Particular mention, however, should be made of N. C. Romero for his untiring programming efforts in writing and maintaining nearly all of the code variations. Also, a special thank you is extended to Juanita Salazar for her excellent job in preparing this manuscript.

REFERENCES

1. Hirt, C. W., Nichols, B. D., and Romero, N. C., "SOLA-A Numerical Solution Algorithm for Transient Fluid Flows," Los Alamos Scientific Laboratory report LA-5852 (1975); LA-5852, Add. (1976).
2. Cloutman, L. D., Hirt, C. W., and Romero, N. C., "SOLA-ICE: A Numerical Solution Algorithm for Transient Compressible Fluid Flows," Los Alamos Scientific Laboratory report, LA-6236 (1976).
3. Harlow, F. H. and Welch, J. E., "Numerical Calculation of Time-Dependent Viscous Incompressible Flow," Phys. Fluids 8, 2182 (1965).
4. Hirt, C. W., "Heuristic Stability Theory for Finite-Difference Equations," J. Comp. Phys. 2, 339 (1968).
5. Viecelli, J. A., "A Computing Method for Incompressible Flows Bounded by Moving Walls," J. Comp. Phys. 8, 119 (1971).
6. Teyssandier, R. G. and Wilson, M. P., "An Analysis of Flow Through Sudden Enlargements in Pipes," J. Fluid Mech. 64, 85 (1974).

7. Nichols, B. D. and Hirt, C. W., "Nonlinear Hydrodynamic Forces on Floating Bodies," Proc. 2nd Intern. Conf. Num. Ship Hydro., September 1977, Berkeley, CA, pp. 382-394.
8. MacCormack, R. W., "Numerical Solution of the Interaction of a Shock Wave with a Laminar Boundary Layer," Proc. 2nd Intern. Conf. Num. Meth. in Fluid Dyn., Springer-Verlag, Berlin, 151 (1970).
9. Hirt, C. W. and Romero, N. C., "SOLA-DF: A Solution Algorithm for Nonequilibrium Two-Phase Flow," Los Alamos Scientific Laboratory report, in preparation.
10. Hirt, C. W., Rivard, W. C., Romero, N. C., Oliphant, T. A., and Torrey, M. D., "SOLA-LOOP: A Non-Equilibrium, Drift-Flux Code for Two-Phase Flow in Networks," Los Alamos Scientific Laboratory report, in preparation.
11. Stein, L. R. and Hirt, C. W., "SOLA-3D: A Solution Algorithm for Transient, Three-Dimensional Fluid Flows," Los Alamos Scientific Laboratory report, in preparation.

This work was performed under the auspices of the United States Department of Energy.

Group T-3
Theoretical Division
University of California
Los Alamos Scientific Laboratory
Los Alamos, NM 87545

INDEX